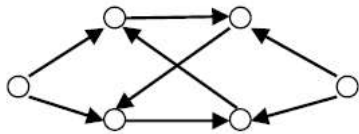


单元测验查看

第七章 图测验

1 下图中的强连通分量的个数为多少个？

How many strongly connected graphs in the under graph?



(填空2 分)

数值精确： 3

解析： 有向图强连通的极大子图称为该有向图的强连通分支或者强连通分量。 分别为最左边1个点组成的极大子图，中间4个点组成的极大子图和最右边1个点组成的极大子图。 Maximal strongly subgraphs of a directed graph are called strongly connected components of this directed graph.They are the subgraph consist of the left-most vertex, the subgraph consist of 4 vertices in the m subgraph consist of the right-most vertex respectively.

2

如果无向图 $G=(V,E)$ 是简单图，并且 $|V|=n>0$ ，那么图 G 最多包含多少条边？

If undirected graph $G=(V,E)$ is simple graph, and $|V|=n>0$, then how many edges can graph G contains at most? (There is only one correct answer)

(填空2 分)

文字精确： $n*(n-1)/2$ 或 $(n-1)*n/2$

3

在一个无向图中，所有顶点的度数之和等于所有边数的()倍。

In a undirected graph, the sum of degrees of all vertices is equal to the amount of all edges times ().

(单选2 分)

☒ A. 2(正确答案)

解析： 由于是无向图，每条边会在这条边连接的两个顶点的度数里都出现一次，故为两倍
Because the graph is undirected, each edge would appear onec in the two vertices which the edge connects.So the answer is 2.

☐ B. 3(错误答案)

☐ C. 1(错误答案)

☐ D. 1/2(错误答案)

4 下面关于图的说法正确的有

The right statements of graphs in the following are:

(多选3 分)

☒ A. 对于无向图，所有结点的度数加起来一定是偶数。 As for undirected graphs, the sum of degrees of all vertices is definitely even number. (正确答案)

解析： 结点度数是边数的2倍，故一定为偶数。
The sum of degrees of vertices is equal to the amount of edge times 2, so it must be even number

☒ B. 将有向图的一个强连通分量中的边全部反向仍然是强连通分量。 Reversion all the edges of a strongly connected component of a directed graph, then the subgraph is still a strongly connected component. (正确答案)

解析： 原来强连通分量中的点必须能够互达，边全部反向后，仍然能够互达。而原来强连通分量外的点和强连通分量内的点之间的边没有变化，以前不能互达现在还是不能，这样是图。
In the original strongly connected component, every pair of vertices in the subgraph is connected by a path.After reversion, this property doesn't change. And the connectivity of the subgraph is not changed.

☐ C. 对于有向图，每个结点的出度必须要等于入度。 As for directed graph, each vertices' out-degree is equal to its in-degree.(错误答案)

解析： 所有结点的出度之和等于入度之和，但是每个结点并没有出度和入度相等的性质。
The sum of in-degrees of all nodes is equal to the sum of out-degrees of all nodes. But for each node, it doesn't work.



☐ D. 对于一个连通图，一定存在一种给边添加方向的方案使得这个图变成强连通图。For a connected graph, there must be a way of directing all the edges of the original graph to make the graph strongly connected graph. (错误答案)

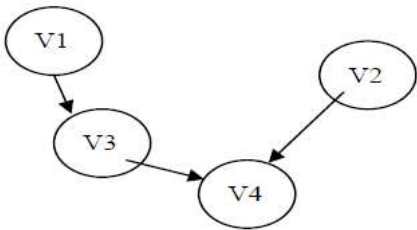
解析：给两个结点新增一条边相连，能够形成一个连通图，但是不管怎么给边定向都不能使其成为强连通图。
Add an edge of two vertex, then we can get a connected graph. But we can't make it strongly connected graph however we direct the edges.

☐ E. 对于有向图，所有结点的入度加起来一定为奇数。For directed graph, the sum of in-degrees of all nodes must be odd number. (错误答案)

解析：只有一个结点构成的有向图，所有结点入度之和为0。
In the case of the graph consist of only one vertex, the sum of in-degrees of all nodes is 0.

5 有向图G如下图所示，请写出所有拓扑排序序列。所有的顶点都直接用其数字标号表示，如拓扑排序序列为 $v_1v_2v_3v_4$ ，那么请写成1234（中间没有空格）。不同的拓扑排序序列按照字典序排序，中间用一个空格隔开。

Directed graph G looks like following graph, please list all the topological order sequences. All the vertices are marked by numbers directly. Like topological order sequence $V_1V_2V_3V_4$, we write it as 1234(with no blank space).Different topological order sequences are sorted according to alphabet order, and separated by a blank space.



（填空2 分）

文字精确：1234 1324 2134

解析：显示解析 根据拓扑排序的定义，顶点1必须在顶点3前，顶点1、顶点2和顶点3必须在顶点4前，故排列可以为1234、1324、2134 According to the definition of topologic order, vertex 1 must appear before vertex 3, vertex 1,2,3 must appear before vertex 4. So the sequences can be 1234, 1324, 2134

6 在有向图G的拓扑序列中，若顶点 V_i 在顶点 V_j 之前，则下列情形不可能出现的是（ ）。
In the topological order sequences of the directed graph G, if vertex V_i appears before V_j , then the impossible situation of the following is ()

（单选2 分）

☐ A. G中有一条从 V_j 到 V_i 的路径 There is a path from V_j to V_i in the G.(正确答案)

解析：如果G中有一条从 V_j 到 V_i 的路径，则顶点 V_j 必须在顶点 V_i 之前
If there is a path from V_j to V_i , V_j must appear before V_i

☐ B. G中有边 (V_i, V_j) G contains edge (V_i, V_j) .(错误答案)

解析：如果G中有从 V_i 到 V_j 的边，顶点 V_i 肯定在顶点 V_j 之前
If G contains a edge from V_i to V_j , vertex V_i must appear before V_j .

☐ C. G中有一条从 V_i 到 V_j 的路径 G contains a path from V_i to V_j .(错误答案)

解析：如果G中有一条从 V_i 到 V_j 的路径，顶点 V_i 肯定在顶点 V_j 之前
If G contains a path from V_i to V_j , V_i must appear before V_j .

☐ D. G中没有边 (V_i, V_j) G doesn't contain edge (V_i, V_j) .(错误答案)

解析：如果G中没有从 V_i 到 V_j 的边，但是G中有一条从 V_i 到 V_j 的路径，顶点 V_i 肯定在顶点 V_j 之前
If G doesn't contain edge (V_i, V_j) , but G contains a path from V_i to V_j , then vertex V_i must appear before V_j .

7 无向图 $G=(V, E)$ ，其中： $V=\{a, b, c, d, e, f\}$, $E=\{(a, b), (a, e), (a, c), (b, e), (c, f), (f, d), (e, d)\}$ ，对该图进行深度优先遍历（优先访问编号小的结点），得到的顶点序列为？

注意：答案中没有空格

Undirected $G = (V,E)$, concretely: $V = \{a,b,c,d,e,f\}$, $E = \{(a,b),(a,e),(a,c),(b,e),(c,f),(f,d),(e,d)\}$,perform depth-first traversal(visit the vertex of small number firstly), what vertices sequence do we get? Notice: no blank space in answer.

（填空2 分）

文字精确：abedfc

解析：根据深度优先的算法，先访问a，再访问a的邻接顶点b，再访问b的邻接顶点e，访问e的邻接顶点d，访问d的邻接顶点f（注意是无向图），访问f的邻接顶点c，不再有没访问的顶点，结束
According to depth-first algorithm, we visit vertex a firstly, then visit its adjacent vertex b, next we visit b's adjacent vertex e, visit e's adjacent vertex d, visit d's adjacent vertex f (notice: its undirected graph), visit f's adjacent vertex c. And there is no vertex not visited, over.

8 当各边上的权值满足什么要求时，宽度优先搜索算法可用来解决单源最短路径问题？

What requirement do the weight of edges should satisfied to make width-first search algorithm can solve single source shortest path problem?

(单选2 分)

- ☒ A. 均相等Equal(正确答案)
- 解析：宽度优先搜索算法的搜索状态树是一层一层的扩展结点的，而当边权均相等时，步数越少距离越短，所以可以直接用宽度优先搜索算法解决。 Width-first search algorithm's search-state-tree expands layer by layer. When the weight of edges is equal to each other, lesser step corresponds to shorter distance, so we can use width-first search algorithm to solve problem.
- ☐ B. 均互不相等 Each edge is not equal to each other.(错误答案)
- 解析：宽度优先搜索算法的搜索状态树是一层一层的扩展结点的，而当边权均相等时，步数越少距离越短，所以可以直接用宽度优先搜索算法解决 Width-first search algorithm's search-state-tree expands vertices layer by layer. When the weight of edges is equal to each other, lesser step corresponds to shorter distance, so we can use wic first search algorithm to solve problem.
- ☐ C. 不一定相等 No limitation.(错误答案)
- 解析：宽度优先搜索算法的搜索状态树是一层一层的扩展结点的，而当边权均相等时，步数越少距离越短，所以可以直接用宽度优先搜索算法解决。 Width-first search algorithm's search-state-tree expands vertices layer by layer. When the weight of edges is equal to each other, lesser step corresponds to shorter distance, so we can use width-first search algorithm to solve problem.
- ☐ D. 不一定相等 No limitation.(错误答案)

9 下列关于最短路算法的说法正确的有：

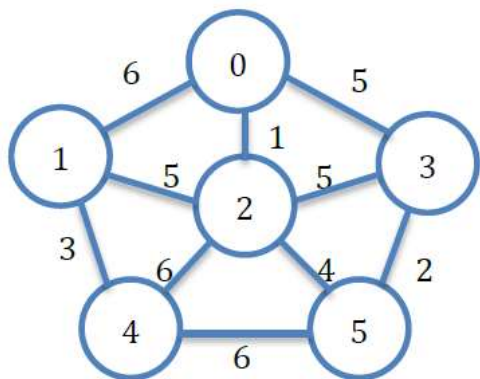
The right statements of the following are:

(多选3 分)

- ☐ A. 当图中不存在负权回路但是存在负权边时，Dijkstra算法不一定能求出源点到所有点的最短路。 When the graph doesn't contain circuit of negative weight, but contains the edge of negative weight. Dijkstra algorithm can't guarantee the correctness of the algorithm. (正确答案)
- 解析：即使是只有负权边，也会导致以前已经被选出来更新其它结点最短路值的结点的最短路值被更新，造成错误。 Even if there is only the edge of negative weight, it would result in that the shortest path of the node which be selected previously to update the shortest path of the other vertices changes, then
- ☐ B. 当图中不存在负权边时，Dijkstra算法能求出每对顶点间最短路径。 When the graph doesn't contain edge of negative weight, Dijkstra algorithm can calculate the shortest path of each pair of vertices. (正确答案)
- 解析：可以执行多次Dijkstra算法实现这一要求。 We can perform Dijkstra algorithm repeatedly to satisfy this requirement.
- ☐ C. 当图中存在负权回路时，Dijkstra算法也一定能求出源点到所有点的最短路。 When the graph contains the circuit of negative weight, Dijkstra algorithm can certainly calculate the shortest path form the single source to all the vertices. (错误答案)
- 解析：Dijkstra算法无法处理图中存在任何负权边的情况。 Dijkstra algorithm can't handle the situation that graph contains any edge of negative weight.
- ☐ D. Dijkstra算法不能用于每对顶点间最短路计算。 Dijkstra algorithm can't be applied to calculate the shortest path of each pair of vertices. (错误答案)
- 解析：可以执行多次Dijkstra算法实现这一要求。 We can perform Dijkstra algorithm repeatedly to satisfy this requirement.

10 请使用Kruskal算法求出下图的最小生成树，依次写出每次被选择的合法的合并代价最小的边的编号（如果同时存在多条边满足要求，选择编号最小的）。顶点a到顶点b (a < b)之间的边编号为ab，例如图中权值为1的边编号为02。(不同编号之间用一个空格分隔)

Please use Kruskal algorithm to the following graph and find the minimum spanning tree, and write the number of the valid vertex with minimum merging cost in turn(if there are many vertices satisfy requirement, choose the vertex with minimum number). The number of the edge connecting vertex a and vertex b is ab. Like the edge with weight 1 in the graph, its number is 02(different numbers separated by a blank space).



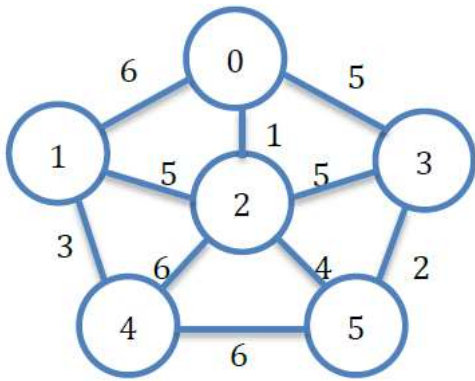
(填空2 分)

文字精确: 02 35 14 25 12

解析: Kruskal算法优先选择权值小的边, 先挑选权值为1的边02, 再选择权值为2的边35, 再选择权值为3的边14, 再选择权值为4的边25, 再选择权值为5的边, 只有选择12才能连接两个不同的
案为02 35 14 25 12 Edges with small weight are given high priority to be chose in the Kruskal algorithm. We firstly choose the edge 02 with weight 1, then choose edge 35 with weight 2, then c
with weight 3, then choose the edge 25 with weight 4, then choose the edge with weight 5, and only the edge 12 can connect two different connected components. So the answer is 02 34 14 25

11 请使用Prim算法从结点0出发求下图的最小生成树, 依次写出每次被加入到最小生成树中边的编号 (如果同时存在多条边满足要求, 选择编号最小的)。顶点a到顶点b (a < b)之间的边编号为ab, 例如图中权值为1的边编号为02。(不同编号之间用一个空格分隔)

Please use prim algorithm starting from vertex 0 to find the minimum spanning tree of the following graph, write the number of the edge added into the minimum spanning tree in turn((if there are many vertices satisfy requirement, choose the vertex with minimum number).
The number of the edge connecting vertex a and vertex b is ab. Like the edge with weight 1 in the graph, its number is 02(different numbers separated by a blank space).



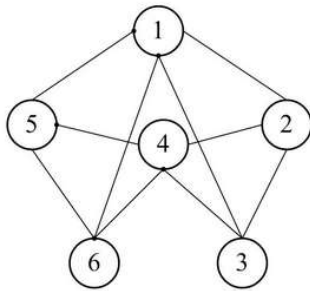
(填空2 分)

文字精确: 02 25 35 12 14

解析: 最小生成树中已经选择的顶点的集合U初始为 {0}, 从0起, 先挑选其他节点到0权值最小为1的边02, 把顶点2加入U, U变为 {0, 2}, 再选择到U权值最小为4的边25, U变为 {0, 2, 5}, 权值最小为2的边35, U变为 {0, 2, 5, 3}, 再选择到U权值最小为5的边12, U变为 {0, 2, 5, 3, 1}, 再选择到U权值最小为3的边14, U变为 {0, 2, 5, 3, 1, 4}, 结束, 答案为02 25
original set of the selected vertices of the minimum spanning tree is {0}. We firstly select the edge with the minimum weight of edges which connect vertex 0 and other vertices. So we select ed
1 and add vertex 2 into U, then U becomes {0, 2}. Next, in the set of edges which connect U and others, we select the edge 25 with the minimum weight 4, then U becomes {0, 2, 5}. Next, in the
which connect U and others, we select the edge 35 with the minimum weight 2, then U becomes {0, 2, 5, 3}. Similarly, then we select the edge 12 with the minimum weight 5, U becomes {0, 2,
select the edge 14 with the minimum weight 3, then U becomes {0, 2, 5, 3, 1, 4}, over. The answer is 02 25 35 12 14

12 题图为一无向图,分别写出从顶点1出发,按深度优先搜索遍历算法得到的顶点序列,和按广度优先搜索遍历算法得到的顶点序列

注意:
1. 优先访问编号小的结点
2. 顶点序列内无空格, 先写深度优先搜索遍历序列, 再写广度优先搜索遍历序列。
3. 深度优先搜索遍历序列和广度优先搜索遍历序列之间用空格隔开
The following graph is an undirected graph, please write the vertices sequences obtained by the depth-first search traversal algorithm and width-
first search traversal algorithm respectively. Notice:
1. The vertex with smaller number should have higher priority to be visited.
2. There is no blank space among the vertices sequences, write the sequence produced by the depth-first search traversal algorithm at first, then
write the another one.
3. There two sequences should be separated by a blank space.



(填空2 分)

文字精确: 123456 123564

解析: 根据深度优先定义, 先访问1, 依次是2、3、4、5、6, 注意是无向图。广度优先是一层一层访问, 即123564, 答案为123456 123564 According to the definition of the dep
then 2, 3, 4, 5, 6 in turn, you should noticed that it's undirected graph. Width-first traversal algorithm visit the vertices layer by layer, the sequence is 123564. So the answer is 12